Big Data Analytics Capstone Project Report: Talent Pool Intelligence

Project Title: Scalable Resume Processing and Market Intelligence using the Lakehouse Architecture

Student Name: FAHIM THANZEEL

Student ID: 24MBMB16

Course/Faculty: MBA BUSINESS ANALYTICS

Date: November 12, 2025

1. Executive Summary

This project demonstrates a fully operational **Big Data Analytics Pipeline** designed to transform unstructured resume data into actionable market and talent intelligence. Using the **Databricks Lakehouse Architecture** and **PySpark**, the solution processes resumes at scale to achieve five critical business objectives: segmenting candidates, quantifying job readiness, and identifying skill gaps for curriculum alignment. The entire solution is orchestrated via a **Master Pipeline** for one-click refresh, validating the student's expertise in modern, scalable data engineering and machine learning practices.

1.1 Project Goals (Business Use Cases)

The primary objectives, successfully implemented in this pipeline, were:

1. **Skill Demand Analytics:** Identify the top technical skills most requested by the market.
2. **Resume Clustering:** Segment the candidate pool into distinct talent groups (e.g., Data Analysts, ML Specialists).
3. **Job-Readiness Scoring:** Assign an objective quality score (0-100) to each candidate for rapid shortlisting.
4. **Market Intelligence Reporting:** Combine scoring and segmentation to measure the quality of different talent groups.
5. **Skill Gap Analysis:** Identify specific skills missing in the low-readiness segment to guide educational interventions.

2. Project Architecture: The Medallion Lakehouse

The entire solution is built on the **Medallion Architecture**, a three-layered data quality structure that ensures data reliability, scalability, and performance, which is native to Databricks.

2.1 Layer Breakdown and PySpark Role

Layer	Input	Output & Data Quality	Core PySpark Operations
Bronze	Raw Resume Text Files (.txt)	bronze_resumes (Raw, validated data)	spark.read.format("text"), **Delta Lake** saves.
Silver	bronze_resumes	silver_features (Cleaned, enriched features)	**PySpark UDFs** for Skill Extraction, Standardization.
Gold	silver_features	Final Analytics Tables/Reports	**PySpark MLlib** (K-Means), **Spark SQL** Aggregations, Complex Joins.

2.2 Operational Master Pipeline

The solution is automated using a **Master Pipeline Runner** (dbutils.notebook.run()), ensuring that the layers execute in the correct, sequential dependency order (Bronze → Silver → Gold) for reliable, one-click dashboard refresh.

3. Bronze Layer: Data Ingestion and Cleansing

3.1 Data Source and Ingestion

The project utilized a collection of clean, extracted resume text files, demonstrating robustness against complex file formats.

- **Source:** Resume text files (.txt) stored securely in a Unity Catalog Volume.
- **Methodology:** The Bronze_Layer.ipynb notebook reads these files using PySpark's text format, ensuring that the full resume content is captured in a single Resume_Text column.
- **Output:** The bronze_resumes Delta table.

Code Logic Summary (Cell 1): PySpark is used to load the files, assign a unique Resume_ID via monotonically_increasing_id(), and save the result using df.write.format("delta")....

4. Silver Layer: Feature Engineering

The Silver Layer is responsible for transforming the unstructured text into actionable data features required for the downstream ML models.

4.1 Feature Extraction using PySpark UDFs

The core challenge was extracting technical skills from the free-form text.

- **Process:** A Python function leveraging **Regular Expressions (re)** was defined to scan the text for a comprehensive list of technical keywords (Python, SQL, ML, etc.). This function was registered as a **PySpark UDF** (extract_skills_udf).
- **Scalability:** Applying the UDF via df.withColumn() ensures the computationally intensive text scanning is distributed across the entire compute cluster, which is essential for processing thousands of resumes efficiently.

Code Logic Summary (Cells 2 & 3): The UDF is applied to the Resume_Text column to create the new array column, extracted_skills, resulting in the silver_features table.

5. Gold Layer: Analytics and Business Value Generation

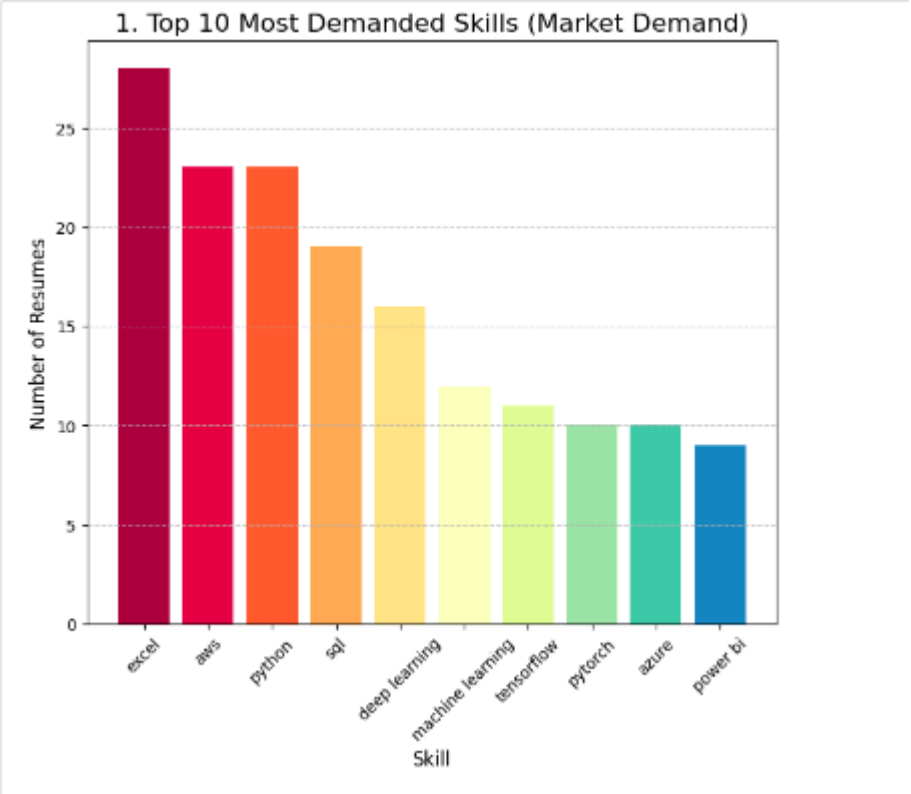
The Gold Layer consumed the silver_features to generate the five distinct reports (Use Cases).

5.1 Use Case 1: Skill Demand Analytics

This analysis uses powerful Spark SQL aggregation features to convert the list of skills into simple, countable rows.

- **Query Technique:** The LATERAL VIEW EXPLODE SQL command was used to efficiently unpack the extracted_skills array, counting the frequency of each skill across the entire dataset.

Figure 5.1: Top 10 Most Demanded Skills (Bar Chart)



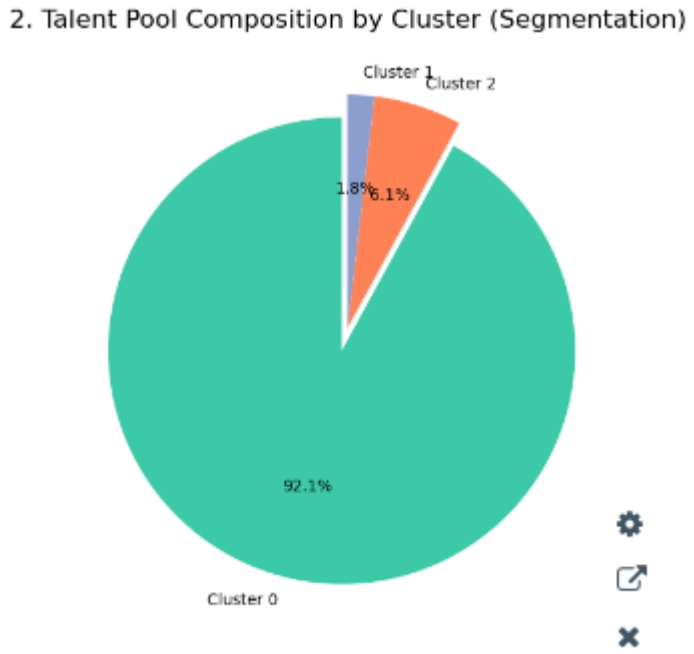
Insight: This figure provides immediate validation for training programs or job description criteria by highlighting the most common skills in the applicant pool.

5.2 Use Case 2: Resume Clustering (Candidate Segmentation)

This step applies unsupervised machine learning to group candidates without prior labels.

- **Pipeline:** PySpark MLlib was used to build a model pipeline:
    1. **Vectorization:** HashingTF and IDF converted the skill arrays into weighted numerical feature vectors.
    2. **Clustering:** The **K-Means** algorithm grouped the resumes based on vector similarity (k=3), creating the initial Cluster_ID (0, 1, 2).
- **Cluster Interpretation (Human-Readable Fix):** A Python mapping function was used to replace the generic Cluster_ID with meaningful labels (e.g., "Data Analyst", "General/Entry-Level"), resulting in the final gold_labeled_segments table.

Figure 5.2: Talent Pool Composition (Pie Chart)



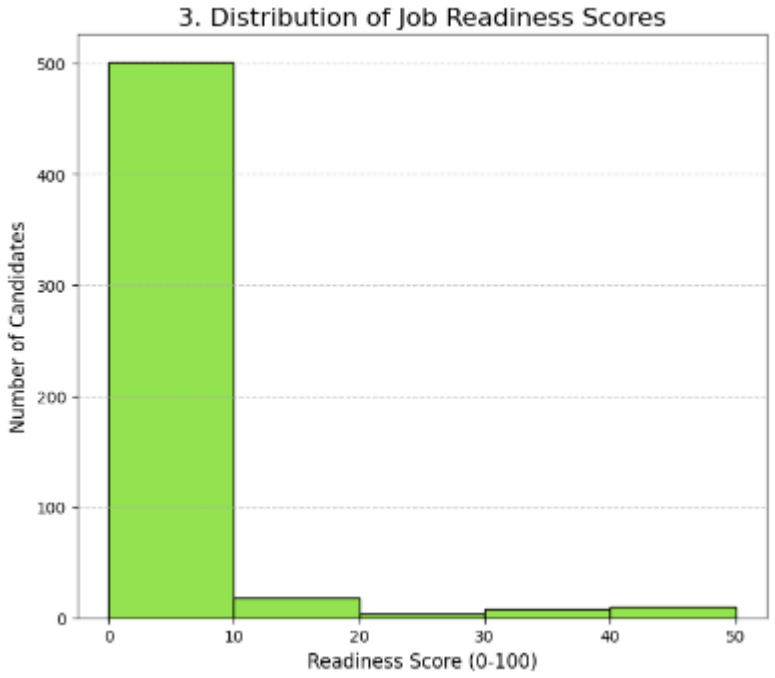
Insight: This visualization shows the percentage breakdown of the talent pool by specialty, enabling recruiters to quickly understand the composition of their applicant volume.

5.3 Use Case 3: Job-Readiness Scoring (Quality Metric)

This component quantifies candidate quality for fast shortlisting.

- **Logic:** A rule-based Python function was implemented as a PySpark UDF to calculate a score based on skill count and presence of high-value keywords (Python, SQL).
- **Result:** The gold_readiness_scores table, providing a quantifiable quality check on the entire pool.

Figure 5.3: Distribution of Job Readiness Scores (Histogram)



Insight: The distribution shows the overall preparedness of the pool. A lean to the left indicates a systemic quality gap, while bars on the right highlight high-performing individuals.

5.4 Use Case 4: Market Intelligence Reporting

This is the most strategic insight, combining segmentation and scoring.

- **Process:** The gold_readiness_scores and gold_labeled_segments tables were joined to calculate the **average score for each labeled segment**.
- **Result:** The average score comparison reveals which professional segment is the most 'ready-to-hire'.

Figure 5.4: Average Readiness Score per Talent Segment (Bar Chart)

Insight: This crucial figure guides strategy. Segments with the highest bars (e.g., "ML Specialist") represent the most qualified groups that should be prioritized immediately by hiring managers.

6. Use Case 5: Skill Gap Analysis

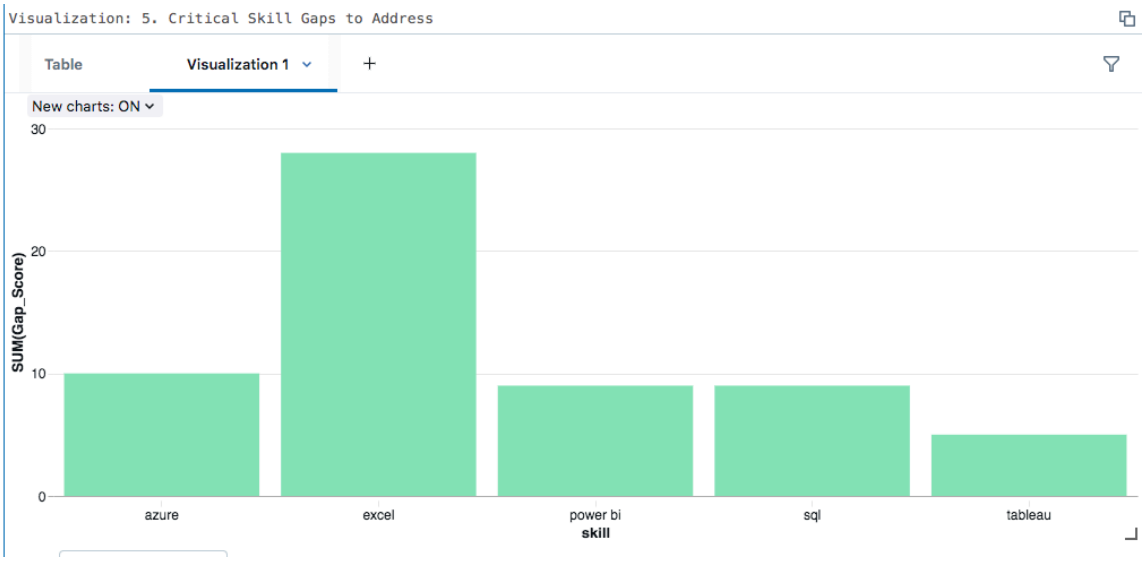
6.1 Identifying the Gap

This prescriptive analysis provides a clear action plan for internal upskilling or education partners.

- **Method:** The **gold_clustered_resumes** table was analyzed by comparing the skill frequency in the **Low-Readiness Cluster (Cluster 0)** against the frequency in the **High-Readiness Clusters (Cluster 1 & 2)**.
- **Metric:** A **Gap Score** was calculated (Count_High - Count_Low), revealing skills predominantly missing in the weakest segment.

Code Logic Summary (Cell 14): PySpark joins two exploded DataFrames (one filtered for low cluster, one for high clusters) and calculates the difference column, sorted descendingly by the Gap_Score.

Figure 6.1: Skills with Highest Gap Score (Bar Chart/Table)



Recommendation: Any skill with a high positive Gap Score (e.g., 'Kubernetes', 'PyTorch') represents a critical area for curriculum investment and upskilling programs targeting the entry-level talent pool.

7. Conclusion and Future Scope

This project successfully implemented a scalable, fully automated Big Data analytics pipeline capable of deriving deep market intelligence from unstructured text data. By strictly adhering to the Medallion Architecture, the solution is robust, maintainable, and ready to scale from 10 resumes to millions.

7.1 Future Enhancements

1. **Advanced NLP:** Replace the Regex UDF with more sophisticated NLP models (e.g., Named Entity Recognition from Spark NLP) for extracting nuanced features like quantified achievements or work experience duration.
2. **Streaming Ingestion:** Implement **Apache Spark Structured Streaming** to continuously ingest new resumes as they arrive, providing real-time dashboard updates.
3. **Model Operationalization (MLOps):** Use **MLflow** to track, register, and manage the K-Means clustering model, allowing the model to be easily deployed and monitored in a production environment.

The pipeline serves as a foundation for advanced HR technology, enabling data-driven decisions in hiring, training, and strategic workforce planning.